INTRODUCTION

What is structure? We know when a computer project or program has good or bad structure, and yet it is difficult to define. Structure can be viewed as an organized, controlled approach to system development. It is the ingredient that makes a system dependable and supportable. Although some structure is needed on a successful computer project, too much can spoil it.

It is almost automatic today to structure large computer systems, but it is not as common to structure Decision Support Systems (DSS), which are increasingly being used to help management make decisions using data from existing computer systems. Management does not have the time to tell the designers what they need, but they want it fast and they are going to change their minds. They say, "I don't know what I want, but I'll know it when I see it."

The real world is a very unstructured environment, and DSS are frequently developed by trial and error and experimentation.

In many ways a prototyping approach using a 4th generation language like SAS is ideal for a DSS, but without structure the system will become uncontrollable. The question is: How much structure should be used in the system design and development of a DSS? Should a prototyping approach be used with minimum structure, or should a modern structured design methodology be used?

To help answer this question I am going to describe a DSS called TRACS that we developed for an oil company using SAS. We wanted an environment which would allow flexibility and creativity, but would also allow us to manage the system development without adding a large amount of overhead. It was decided to use a prototyping approach, but add some structure to control the system. The main point of this paper is that you have to devote some time to structuring a DSS to have a dependable, responsive system.

PROJECT DESCRIPTION

The TRACS project began to help management make strategic decisions on which markets to expand or contract. TRACS is an economic model which determines the distribution and financial costs of marketing gasoline and fuel oil products.

Figure 1 shows the distribution system of a typical refining and marketing oil company. The TRACS system computes the marketing cost using information extracted from 6 computer systems. The major cost components are:

- Product manufacturing cost
- Transportation cost
- Terminal storage and handling
- Credit card costs

A major design objective of the system was to also use the time value of money in determining the cost of doing business. Essentially this involved determining how long product was tied up in inventory and transportation, and how long it took to receive cash from customers. Internal cost of capital is used to determine the cost of:

- Transportation time
- Inventory
- Customer receivables
- Exchange receivables
- Exchange product payback

A second major objective was to determine the cost of using exchange agreements with other oil companies. An exchange is used in the oil business to reduce transportation and terminal costs. A company gives an exchanger product at its terminal in exchange for product at a city which is further from the refinery. The TRACS system values the cost of exchange balances and the timing and cost of exchange paybacks to determine the cost of exchange activity. This turned out to be a complex problem because exchanges are seldom in balance.

After costing, the data is stored in a SAS database which has volumes, revenues, cost, and profit by year, month, product, terminal, class of business, exchange company, and customer. The user then runs analytical reports and plots to help make strategic decisions. Figure 2 shows the general design of the system.
NEED FOR STRUCTURE

This system began as a 6 week feasibility study. When the results of the study were encouraging, it was decided to do a prototype of the system. It became evident rather quickly that some structure was needed on this project. Since I was an outside consultant, the users needed some assurance that the system could operate after I left. Also, the system promised to be very complex to design, program, and run. I felt that if this system was going to become a management tool, it had to be developed with some structure.

Structure is required simply to assist the users and analysts remember details about the system design. Without a consistent, logical approach a large system becomes unmanageable. Some of the structuring techniques you can use are top-down design, data flow diagrams, user review, data dictionaries, menus, and documentation. The approach can be as simple as a design philosophy and a filing cabinet, or as complex as one of the large project management methodologies.

STRUCTURED VERSUS PROTOTYPING

Frequently one finds two groups in large organizations. One group is responsible for developing major business systems. They are usually called the DP or MIS department and they are increasingly adopting structured approaches to project management. The other group (called Management Science, Technical Systems, or something similar) was formed to develop small projects quickly, and they favor a prototyping approach. This second group is where SAS is frequently used.

I have worked in both of these groups and I prefer the prototyping approach, but there are some problems. Prototyping is typically very unstructured. The main goal is to deliver a result fast, and structured procedures slow the development and are unwelcome. However, my observation has been that pure prototyping projects without structure are not easily converted to production systems. They are more suitable for small or one time only projects.

On the other hand, projects developed using structured methodologies become large and cumbersome. Structure adds new complexity to the system, and the cost of the structure can easily exceed the cost of design and programming. The system becomes difficult to change; speed, flexibility, and creativity are reduced. Structured methodologies are more suitable for large systems that require dependability.

The key question is: How much structure should be used on a DSS developed in a 4th Generation Language? Obviously, a DSS will lie somewhere between a prototyping and structured approach. Figure 3 shows typical time allocations for different type projects. The amount of time allocated to structure will vary depending on the requirements of the system for:

- Dependability
- Accuracy
- Flexibility
- Longevity
- Security

One must weigh the advantages of control, documentation, and easier support against the cost of using structured procedures. The existing installation standards will also have an impact. However, there is no pat formula for how much time should be spent on structure.

STRUCTURED APPROACH USED ON TRACS

In the TRACS project the client wanted to develop a useful system quickly, but we also wanted a responsive system which could be expanded. Since the system was going to be used to make major decisions, we also wanted accurate, reliable data. Therefore, a blend of prototyping and structured philosophies was used.

Prototyping concepts used were:
1. Use a 4GL (SAS) to analyze data and help design the system.
2. Give users a sample report from actual data so they can plan report requirements.
3. Keep the system flexible so the user can change the design easily.

Structured concepts used were:
1. Document during development.
2. Get frequent user feedback and approval during development.
3. Provide for reruns, and security.
4. Use menus for system operation, testing, and user reporting.
5. Provide a non-technical reporting interface to the user.
6. Use a consistent, disciplined, structured approach to programming.
PROJECT MANAGEMENT TECHNIQUES USED.

We felt that the key to managing the system was to have an interactive menu to run the system. We also wanted to give the user of the system the capability to vary the variables used on an analysis. However, my client did not have the SAS/FSP* or SAS/AF* product, therefore we developed menus using the IBM CLIST and EDIT programs. A CLIST is an IBM program which interacts with a TSO user and issues TSO commands or submits jobs under CLIST control. Figure 4 shows the way the system was structured for development, testing and use.

The CLIST works as follows:
1. The user selects an option from main and sub menus.
2. The user is prompted for variables for the option selected.
3. The JCL and program requested are copied to a work dataset.
4. The CLIST uses the TSO EDIT program to change the variables to the ones the user requested.
5. The CLIST submits the changed program for batch execution.

Variables changed by the user include program name, datasets, time periods, selection criteria, sort fields, and summarization fields. The SAS %INCLUDE command was sometimes used to read the parameters into the program from a PARM file. We had originally intended to use MACROS, but the above approach handled all our requirements very simply; therefore MACROS were only used for plots. Figure 5 shows the main menus that were used for the programmer or user.

This framework provided better control of the project. The programmer used the same menus that a user would, but specified that a test database or test program be used. It was a simple matter after testing to replace the production program with the new version.

The user was also presented with a friendly report menu. He could run more than 40 programs, varying the time period being selected for study, or the fields used for sorting and totaling reports. Some reports were grouped to allow study of special factors such as state, customer, etc. The user could also select data and prepare plots and replay the plots online or print them on a plotter. The main advantage of the menu was that the system was easy to use in a consistent and flexible manner.

CONCLUSION

This approach proved to be easy to use and provided flexibility and structure to the system. The system was developed in less than 1 man year, and is being used presently to make important decisions. I feel that a similar system developed using conventional methods would have taken 3 man years to do. The use of SAS and a structured approach were essential ingredients of a successful system. There is no doubt in my mind that if time had not been spent on structuring the system, it would have been impossible to use to provide reliable and flexible response to users.

I estimate that about 25% of the development cost was for menus, documentation, and other structuring methods. Be sure that you allow enough time in your estimates to properly structure your systems if they will be used to make major decisions. There will be a lot of pressure on you to develop a system fast, but it is the designer's responsibility to insist on the time to properly structure the system.

* SAS, SAS/FSP, and SAS/AF are registered trademarks of SAS Institute Inc., Cary, NC, USA.

REFERENCES


For more information contact:
John Stuart Consulting
6345 S. 70th E. Ave.
Tulsa, OK 74133
918-494-9548